07

Controlando checked exceptions

Transcripción

[00:00] Bienvenidos nuevamente. Para dejar este concepto aún más afianzado, vamos a repetir el mismo ejercicio que hicimos en nuestro proyecto actual del curso, pila de ejecución. ¿A qué me refiero? Voy a llegar aquí a saldo insuficiente exception y voy a obligar a que esta excepción sea tratada. ¿Cómo consigo eso?

[00:20] Extendiendo directamente de excepción para que sea una exception del tipo checked, verificado. Voy a guardar aquí y automáticamente vemos que dejó de compilar aquí. ¿Por qué? Porque él está lanzando esa excepción pero yo en ningún momento estoy avisando que este método va a lanzar esta excepción, ¿entonces cuál es la única solución que yo tengo aquí?

[00:47] Añadir la declaración throws SaldoInsuficienteException, entonces yo aquí estoy alertando, este método saca lanza esta bomba. El método saca va a lanzar la bomba , del tipo SaldoInsuficienteException entonces quienquiera que llame a este método, prepárate porque vas a recibir una bomba del tipo SaldoInsuficienteException. ¿Y quién fue el primer afectado? Transfiere.

[01:13] ¿Por qué? Porque transfiere llama al método saca. Y ya vimos, diversas formas en las que yo puedo atrapar este error. Puede ser simplemente diciendo: "por si acaso, este método también va a lanzar un SaldoInsuficienteException". Y paso el problema para un nivel más arriba. O si no quiero hacer eso, lo que puedo hacer es simplemente un try/catch.

[01:43] Y quedaría así, simplemente yo no necesito avisarle al método que me va a llamar, que va a recibir una bomba del tipo de SaldoInsuficienteException, porque yo ya estoy atrapando esa bomba aquí y esa bomba ya no va a subir más en la pila de ejecución, sino simplemente va a quedar ahí.

[02:05] Y en el caso de cuenta corriente también tenemos otro error. ¿Y por qué? Porque aquí el método saca tampoco está siendo atrapado. ¿Entonces yo qué necesito hacer aquí? También tratar este método, perdón, tratar este error, tratar esta excepción, y lo puedo hacer de la misma forma.

[02:24] Lo que voy a hacer aquí, es decirle que en la firma del método throws SaldoInsuficienteException, y listo, él va a seguir compilando tranquilamente. ¿Por qué? Porque él en este punto está lanzando el exception y no hay quién lo trate hasta aquí. Y ahora el último error que tenemos y es aquí en nuestra clase que hemos creado anteriormente, TestCuentaExceptionSaldo. ¿Por qué?

[02:52] Porque él aquí está efectuando la operación saca, la cual va a detonar esa bomba, y el método main no está haciendo tratamiento de esa excepción. Entonces, nuevamente, ¿qué caminos tengo yo para hacer tratamiento de la excepción? Fácil. Puedo decirle simplemente a throws declaration.

[03:15] Entonces, el método main ahora va a tener en la firma throws SaldoInsuficienteException, lo cual de por sí ya es una muy mala práctica, porque recordemos que el método main es el método principal de nuestro sistema. Entonces, si el método main lanza un SaldoInsuficienteException, ¿quién va a tratar esa excepción? Nuestra aplicación, simplemente va a morir en el acto, si esa excepción fuera tratada.

[03:43] ¿Entonces qué necesito hacer yo? Tratarla con otro tipo de estructura llamado try/catch. ¿Cómo hago eso? Le puedo decir directamente que el IDE me ayude con eso y él va a hacer aquí un try/catch. En este caso, como yo lo hice seleccionando pues una sola línea de código y yo puse dos para el fin del

ejemplo, el que está en la llamada al método que está fuera de la estructura de try/catch, va a seguir dando el error. ¿Por qué?

[04:16] Porque porque aquí vamos a ver que el scope, el alcance del try, el contexto del try está aquí nada más, entre estas dos llaves, entonces try va a desactivar todo lo que esté entre sus llaves y catch va a atrapar la excepción que yo le diga aquí, pero si yo tengo esta llamada fuera del contexto del try, entonces try no tiene ningún control sobre esto de aquí, sobre esta línea de aquí. ¿Qué puedo hacer yo?

[04:49] Lo meto dentro del try, entonces en el try, ahí ya yo tengo mi paraguas para estas dos líneas de código, quien sea que lance la excepción, sea esta, o sea esta de aquí, try ya me da ese paraguas para estar protegido de este error y atrapar esta excepción del tipo checked, verificada y ahora el compilador ya no me está reclamando nada. ¿Por qué?

[05:18] Porque yo ya estoy haciendo el tratamiento de esta excepción. Entonces, nuevamente. El módulo de excepciones es un poco complicado de entender al inicio, pero si ustedes van practicando crear sus propios errores, crear sus propias bombas, por así decirlo e ir añadiendo eso a su flujo de control de errores, como lo hemos hecho aquí en la clase cuenta.

[05:49] Que si es que yo no tengo saldo, entonces es saldo insuficiente, si mi número de cuenta no fuera válido, entonces tengo otra excepción tipo cuenta no válida, excepción y cosas así, n cosas, entonces sí me gustaría que sigan practicando este tema de los errores, que practiquen bastante cambiando aquí el tipo de excepción, si va a ser checked o unchecked, porque ahí ustedes ya van a comenzar a agarrar la onda más o menos de qué trata.

[06:24] En qué casos les conviene usar un tipo, en qué casos les conviene usar otro tipo. Y recuerden pues que hasta las mismas exceptions de Java extienen de esas dos clases, van a entender hasta cómo funciona la JVM por dentro, por

qué es que en algunos casos lanza checked, por qué es que otras excepciones son unchecked.

[06:40] Hay toda una explicación y toda una ciencia debajo de eso. Entonces dejen sus dudas en el foro, yo voy a intentar responderlas siempre lo más rápido posible. Y nos vemos en la siguiente clase. Chau.